Introduction to Quantum Computing
Lecture 2: Quantum Algorithms

Heather M. Gray

Many thanks to Umesh Vazirani and Irfan Siddiqi for material used in these slides

CERN Academic Training, March 2021
Recap from Monday

Qubits
Quantum Gates
Quantum Circuits
Qubit Technologies
Outline for the lectures

• **Lecture 1: Fundamentals**
  • A brief history, qubits, quantum circuits, qubit technologies

• **Lecture 2: Quantum computers and quantum algorithms**
  • Quantum computers today, quantum algorithms, error correction, quantum advantage

• **Lecture 3: Applications of quantum computing in HEP**
  • Applications of quantum computing to HEP: simulation, reconstruction and physics analysis; including quantum machine learning
Further Reading

• Online resources
  • Scott Aaronson's blog
  • Umesh Varizani's introductory course, graduate
  • John Preskill's graduate course, lecture notes, youtube playlist

• Books
  • Nielsen and Chuang (very complete)
  • Stolze, Suter (introductory, physics oriented)
  • Aaronson (popular)
  • Wilde (graduate level)
  • Watrous (mathematical)
  • Kitaev, Shen and Vyalyi (advanced)
Day 2 Outline

- Quantum computers today
- Quantum Algorithms (Review article)
  - Quantum Fourier transform
  - Shor’s Factoring Algorithm
  - Quantum Search (Grover)
- Quantum Programming Languages
- Quantum Error Correction
- Quantum Simulation
- Quantum Advantage

A lightening-fast overview of a range of topics
Quantum Computers Today
Quantum Computers Today

Noisy Intermediate-Scale Quantum (NISQ) Era

IBM Quantum
Hummingbird (65 qubits)

D Wave
Avantage
5000+ qubits

USTC Jiuzhang

Google Sycamore
53 qubits

Credit: Connie Zhou for IBM

Image Credit: DWave

Forest Stearns, Google AI Quantum Artist in Residence
Erik Lucero, Research Scientist and Lead Production Quantum Hardware
## Current Quantum Computers

>193 quantum computing startups worldwide

### Circuit-based Quantum Computers

These QPUs are based on the quantum circuit and quantum logic gate-based model of computing.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Name/Codename/Designation</th>
<th>Architecture</th>
<th>Layout</th>
<th>Socket</th>
<th>Fidelity</th>
<th>Qubits</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>USTC</td>
<td>Jiuzhang</td>
<td>Photonics</td>
<td>N/A N/A</td>
<td>N/A</td>
<td>76 qubits [8][9][6]</td>
<td>2020</td>
<td></td>
</tr>
<tr>
<td>Google</td>
<td>Bristlecone</td>
<td>Superconducting</td>
<td>6x12 lattice</td>
<td>N/A</td>
<td>99% (readout) 99.9% (1 qubit) 99.4% (2 qubits)</td>
<td>72 qubits [3][4]</td>
<td>March 5, 2018</td>
</tr>
<tr>
<td>Google</td>
<td>Sycamore</td>
<td>Nonlinear superconducting resonator</td>
<td>9x6 lattice</td>
<td>N/A</td>
<td>54 transmon qubit 53 qubit effective</td>
<td>2019</td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>IBM Q 53</td>
<td>Superconducting</td>
<td>N/A N/A</td>
<td>N/A</td>
<td>53 qubits</td>
<td>October 2019</td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>IBM Q 50 prototype</td>
<td>Superconducting</td>
<td>N/A N/A</td>
<td>N/A</td>
<td>50 qubit [10]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Google</td>
<td>N/A</td>
<td>Superconducting</td>
<td>7x7 lattice</td>
<td>N/A</td>
<td>99.7% [1]</td>
<td>49 qubits [2]</td>
<td>Q4 2017 (planned)</td>
</tr>
<tr>
<td>Intel</td>
<td>Tangle Lake</td>
<td>Superconducting</td>
<td>N/A 108-pin cross gap</td>
<td>N/A</td>
<td>49 qubits [13]</td>
<td>January 9, 2018</td>
<td></td>
</tr>
</tbody>
</table>

### Quantum Annealers

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Name/Codename/Designation</th>
<th>Architecture</th>
<th>Layout</th>
<th>Socket</th>
<th>Fidelity</th>
<th>Qubits</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-Wave</td>
<td>D-Wave Advantage</td>
<td>Superconducting</td>
<td>Pegasus P16 [19][20]</td>
<td>N/A</td>
<td>N/A</td>
<td>5000 qubits</td>
<td>2020</td>
</tr>
<tr>
<td>D-Wave</td>
<td>D-Wave 2000Q</td>
<td>Superconducting</td>
<td>C_{16} = Chimera(16,16,4) [17] = 16x16 K_{4,4}</td>
<td>N/A</td>
<td>N/A</td>
<td>2048 qubits</td>
<td>2017</td>
</tr>
<tr>
<td>D-Wave</td>
<td>D-Wave 2X</td>
<td>Superconducting</td>
<td>C_{12} = Chimera(12,12,4) [17][18] = 12x12 K_{4,4}</td>
<td>N/A</td>
<td>N/A</td>
<td>1152 qubits</td>
<td>2015</td>
</tr>
<tr>
<td>D-Wave</td>
<td>D-Wave Two</td>
<td>Superconducting</td>
<td>C_{8} = Chimera(8,8,4) [17] = 8x8 K_{4,4}</td>
<td>N/A</td>
<td>N/A</td>
<td>512 qubits</td>
<td>2013</td>
</tr>
<tr>
<td>D-Wave</td>
<td>D-Wave One (Ranier)</td>
<td>Superconducting</td>
<td>C_{4} = Chimera(4,4,4) [17] = 4x4 K_{4,4}</td>
<td>N/A</td>
<td>N/A</td>
<td>128 qubits</td>
<td>11 May 2011</td>
</tr>
</tbody>
</table>
Google Sycamore and IBM

- IBM and Google use transmon superconducting qubits
- Sycamore has a 2D array of 54 transmon qubits (53 functional)
  - Each qubit is coupled to four other qubits
  - Aluminium for metallization and Josephson junctions; indium for bump-bonds between two silicon wafers
- Operates at 20mK in a dilution refrigerator
- More recent chip from IBM is Hummingbird with 65 qubits
  - “features 8:1 readout”
- IBM has stated that they plan to double the number of qubits available each year
  - And all are named after birds
• Optical quantum computer (photons)
• Light sources; beam splitters; mirrors, 100 photon detectors
  • Use repeated splitting and merging to obtain interference
• Custom-built (non-programmable) quantum computer designed to demonstrate quantum advantage
• Programmable photonic computer (8 qubits) from Xanadu
More about Trapped Ion Computers

- **Qubits**: Ground and excited state or two ground state hyperfine levels (laser cooled to reduce noise)
- **Initialization**: Optical pumping with a laser (fidelity > 99.9%)
- **Single qubit operations**: Drive atomic states with resonant optical or microwave fields/Raman transition
- **Two qubit operations**: Charge interaction between trapped ions
  - Excite on ion to oscillate and induce the second to move
  - Carefully tune the field frequency to push the ions only if the qubit is in a specific state
  - Maintain coherent excited motion
- **Measurement**:
  - Laser applied to ion that couples only to one qubit state
  - Photomultiplier tubes to collect emitted photons
- **More details or here**
Honeywell and IonQ

• Current performance of trapped ion technologies
  • Single qubit coherence of 1 hour
  • High single and double qubit gate fidelity
  • Full connectivity

• IonQ and Honeywell produce quantum computers using trapped ion systems of Ytterbium (\(^{171}\text{Yb}^+\)) ions
  • Most recent is 32 qubits (IonQ) and 10 qubits (Honeywell)
  • Extremely impressive quantum volume

More details, 2, 3
Aside: Comparing Quantum Computers

• Comparing the performance of quantum computers today is tricky

• Many different technologies are used to qubits, so computers with the same number of qubits can have dramatically different performance
  • Qubit lifetime
  • Gate fidelity
  • Gate operation time
  • Connectivity

• Paper comparing the performance of different systems

• One metric that has been proposed is the quantum volume (from IBM)
  • Represents the maximum size of square quantum circuits

• IBM: quantum volume of 64 (27 qubit)

• Honeywell: quantum volume of 512

• IonQ: quantum volume of 4 million
Quantum Algorithms
Quantum Zoo

This is a comprehensive catalog of quantum algorithms. If you notice any errors or omissions, please email me at stephen.jordan@microsoft.com. (Alternatively, you may submit a pull request to the repository on github.) Your help is appreciated and will be acknowledged.

Quantum Algorithms

It’s HARD to develop a new quantum algorithm

Algebraic and Number Theoretic Algorithms

Algorithm: Factoring
Speedup: Superpolynomial
Description: Given an n-bit integer, find the prime factorization. The quantum algorithm of Peter Shor solves this in \( O(n^3) \) time [82, 125]. The fastest known classical algorithm for integer factorization is the general number field sieve, which is believed to run in time \( 2^{\tilde{O}(n^{1/3})} \). The best rigorously proven upper bound on the classical complexity of factoring is \( O(2^{n/4+\gamma(1)}) \) via the Pollard-Strassen algorithm [252, 362]. Shor's factoring algorithm breaks RSA public-key encryption and the closely related quantum algorithms for discrete logarithms break the DSA and ECDSA digital signature schemes and the Diffie-Hellman key-exchange protocol. A quantum algorithm even faster than Shor’s for the special case of factoring “semiprimes”, which are widely used in cryptography, is given in [271]. If small factors exist, Shor's algorithm can be beaten by a quantum algorithm using Grover search to speed up the elliptic curve factorization method [366]. Additional optimized versions of Shor's algorithm are given in [384, 386]. There are proposed classical public-key cryptosystems not believed to be broken by quantum algorithms, cf. [248]. At the core of Shor's factoring algorithm is order finding, which can be reduced to the Abelian hidden subgroup problem, which is solved using the quantum Fourier transform. A number of other problems are known to reduce to integer factorization including the membership problem for matrix groups over fields of odd order [253], and certain diophantine problems relevant to the synthesis of quantum circuits [254].

Algorithm: Discrete-log
Speedup: Superpolynomial
Description: We are given three n-bit numbers a, b, and N, with the promise that \( b = a^s \mod N \) for some s. The task is to find s. As shown by Shor [82], this can be achieved on a quantum computer in poly(n) time. The fastest known classical algorithm requires time superpolynomial in n. By similar techniques to those in [82], quantum computers can solve the discrete logarithm problem on elliptic curves, thereby breaking elliptic curve cryptography [109, 14]. A further optimization to Shor’s

Navigation

Algebraic & Number Theoretic
Oracular
Approximation and Simulation
Optimization, Numerics, & Machine Learning
Acknowledgments
References

Translations

This page has been translated into:
Japanese
Chinese

Other Surveys

For overviews of quantum algorithms I recommend:
Nielsen and Chuang
Childs
Preskill
Mosca
Childs and van Dam
van Dam and Sasaki
Bacon and van Dam
Loceff
Montanaro
Hidary
Quantum Promise

- Certain problems which are difficult classically, are easy on quantum computers
  - e.g. factoring with its superpolynomial speedup
- Can’t efficiently simulate a quantum computer on a classical computer
  - Expected to be hard due to complexity arguments
Algorithmic Complexity

$P = NP$?

Are $P$ and $BQP$ disjoint?

Does $BQP$ extend beyond $NP$?

Reminder: $O(f(n))$ is the maximum time given input of size $n$
Why Quantum Algorithms?

• Quantum Promise
  • Exponential speedup
  • Pattern recognition/Fourier analysis
  • Efficient Searches
  • Minima finding
  • Matrix mathematics: machine learning, linear algebra, etc

• Quantum Challenge
  • Decoherence: short lifetimes of qubits
  • Noise
  • Significant classical resources are often required to interface quantum computers
    • e.g. I/O, error correction
Examples of Quantum Algorithms
Quantum Fourier Transform

- Quantum implementation of the inverse discrete Fourier transform
  - $|\alpha\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \cdots + \alpha_n |n\rangle \rightarrow |\beta\rangle = \beta_0 |0\rangle + \beta_1 |1\rangle + \cdots + \beta_n |n\rangle$
- Exponentially faster than the Fast Fourier transform (FFT): $O(n2^n) \rightarrow O(n^2)$
- Only measure one of the $n$ components
  - also known as quantum Fourier sampling
- Generalization of the Hadamard transformation through the addition of phase

$$QFT_M = \frac{1}{\sqrt{M}} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{M-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2M-2} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3M-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{M-1} & \omega^{2M-2} & \omega^{3M-3} & \cdots & \omega^{(M-1)(M-1)} \end{pmatrix}$$

$\omega^i = \text{complex roots of unity}$
Quantum Fourier Transform

- Circuit implementation uses the Hadamard (H) and controlled phase ($R_m$) gates

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{and} \quad R_m = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^m}} \end{pmatrix}$$
Shor’s Algorithm

• An early quantum algorithm which stimulated significant interest in quantum algorithms is Shor's algorithm for factorization

• Given an integer \( N = p \times q \) for primes \( p \) and \( q \), determine \( p \) and \( q \)

• Best classical algorithm has an execution time of \( \exp(O(\log N)^{1/3})(\log \log N)^{2/3} \), while Shor’s algorithm: \( O(\log N)^3 \)

• Many currently used cryptography algorithms, e.g. RSA public-key, rely on the factorization of large integers being extremely difficult

• Classically: 768 bit number factorized with hundreds of computers over 2 years: \( \sim 10^{20} \) operations

• Quantum: 2000 bit number could be factorized by a billion qubit and \( 3 \times 10^{11} \) gate quantum computer in just over a day
Shor's Algorithm: Implementation

- Shor’s algorithm reduces factorization to a special case of the hidden subgroup problem (HSP)

- More general solutions to HSP (or for other groups) would imply that other cryptographical systems would be broken

- Shor's algorithm consists of two parts:
  - Reduce the factoring problem to the problem of order-finding (can be done classically)
  - Quantum algorithm to solve the order-finding problem
Grover’s Search

- Grover's algorithm targets searching for a specific element within an unsorted database
  - Uniform superposition of over all possible solutions
  - Destructively interfere states that are not solutions (repeat)
- Classical solutions: $O(N)$
- Grover’s algorithm: $O(\sqrt{N})$
- Extended into Grover-Long with zero failure rate
Applications of Grover’s Algorithm

• Grover’s algorithm is very versatile and can also be used as part of more complicated quantum algorithms

• Examples include
  • Finding the minimum of an unsorted list of N integers (quadratic speedup)
  • Determining graph connectivity \( (O(N^2) \rightarrow O(N^{3/2}) \)
  • Pattern matching, i.e. find a pattern \( P \) of length \( M \) within a text \( T \) of length \( N \) \( (O(N + M) \rightarrow O(\sqrt{N} + \sqrt{M}) \)
Quantum inspired classical algorithms

- Sample preference matrix, $T_{ij}$, for users (i) and products (j)
- Low rank approximation
- Generate suggestions for users
- $T(n \times m$ matrix): $O(poly(k)$, $poly (mn)]$; Reduced rank k

A quantum-inspired classical algorithm for recommendation systems

Ewin Tang
May 10, 2019

Abstract

We give a classical analogue to Kerenidis and Prakash’s quantum recommendation system, previously believed to be one of the strongest candidates for provably exponential speedups in quantum machine learning. Our main result is an algorithm that, given an $m \times n$ matrix in a data structure supporting certain $\ell^2$-norm sampling operations, outputs an $\ell^2$-norm sample from a rank-$k$ approximation of that matrix in time $O(poly(k) log(mn))$, only polynomially slower than the quantum algorithm. As a consequence, Kerenidis and Prakash’s algorithm does not in fact give an exponential speedup over classical algorithms. Further, under strong inut assumptions, the clas-


Challenge: Prove this is the best algorithm
Result: A better classical algorithm

Also: Genetic algorithms
Programming Quantum Computers
Quantum Programming Languages

• The field is still at a very early stage of development
  • In some senses, more similar to programming in assembler or for hardware than in modern programming languages like python or C++
  • You need to program the quantum gates for the circuit explicitly
• Essentially every vendor has their own programming language or software development kit (SDK)
  • In many cases the interface is via python
  • Some of them are at rather early stages of development, so regular debugging is required.
• Some examples
  • Qiskit (IBM)
  • Cirq (Google)
  • Ocean SDK (D-Wave)
  • Forest SDK, using Quil (Rigetti)
Simple Examples

```python
import cirq

# Pick a qubit.
qubit = cirq.GridQubit(0, 0)

# Create a circuit
circuit = cirq.Circuit(
    cirq.X(qubit)**0.5,  # Square root of NOT.
    cirq.measure(qubit, key='m')  # Measurement.
)
print("Circuit:")
print(circuit)

# Simulate the circuit several times.
simulator = cirq.Simulator()
result = simulator.run(circuit, repetitions=20)
print("Results:")
print(result)
```

Similar structures
Also, in many cases, simple to get a small amount of time to run on quantum hardware
Generally possible to register for a free account

```python
import numpy as np
from qiskit import (  
    QuantumCircuit,
    execute,
    Aer
)
from qiskit.visualization import plot_histogram

# Use Aer's qasm_simulator
simulator = Aer.get_backend('qasm_simulator')

# Create a Quantum Circuit acting on the q register
q_circuit = QuantumCircuit(2, 2)

# Add a H gate on qubit 0
q_circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
q_circuit.cx(0, 1)

# Map the quantum measurement to the classical bits
q_circuit.measure([0, 1], [0, 1])

# Execute the circuit on the qasm simulator
job = execute(q_circuit, simulator, shots=1000)

# Grab results from the job
result = job.result()

# Returns counts
counts = result.get_counts(q_circuit)
print("\nTotal count for 00 and 11 are: ", counts)

# Draw the circuit
q_circuit.draw()
```

Total count for 00 and 11 are: {'00': 498, '11': 512}
Quantum Error Correction
Quantum Error Correction

- Currently available quantum computers are extremely noisy (NISQ)
  - High potential for errors to occur during calculations
- Somewhat a Catch22: noise is induced through outside interaction, yet we need to interact with the quantum computer for input and output
- For classical algorithms, we typically use redundancy to mitigate errors
  - Not possible for quantum algorithms due to the “no clone” theorem
  - Instead, need to spread the information on a single qubit onto a number of qubits

Simple example: single bit flip code
More complex: Shor’s correcting code

Corrects for bit flip, phase flip or both, Ref1, Error Correction Tutorial
Other Methods for Error Correction

- Error correction for bosonic systems
  - GKP [arXiv:0008040]
- Topological quantum computing

Fault-tolerant quantum computation by anyons

A. Yu. Kitaev

L.D. Landau Institute for Theoretical Physics,
117940, Kosygina St. 2

e-mail: kitaev@itp.ac.ru

February 1, 2008

Abstract

A two-dimensional quantum system with anyonic excitations can be considered as a quantum computer. Unitary transformations can be performed by moving the excitations around each other. Measurements can be performed by joining excitations in pairs and observing the result of fusion. Such computation is fault-tolerant by its physical nature.
Quantum Simulation
Quantum Simulation

• One of the original motivating ideas behind quantum computers was to be able to simulate quantum systems

  • *Nature isn’t classical . . . and if you want to make a simulation of Nature, you’d better make it quantum mechanical*, Richard Feynman, 1981

• Potentially likely to be one of the earlier applications of quantum computers

• Can imagine quantum simulations for quantum chemistry, superconductivity, metamaterials and high-energy physics (see tomorrow’s lecture)

• Typically, what we have in mind is calculating the dynamical properties of the system from the Hamiltonian using the Schrodinger equation

• Can envision using a large digital quantum computer for such simulations, but could also use an analog quantum computer to use one quantum system to mimic another

*Overview of quantum simulation*
Quantum Simulation Examples

Beryllium hydroxide molecule

Water molecule on trapped ion QC


arxiv: 1902.10171.pdf

PRX 6, 031007 (2016)

Hydrogen using VQE
Diazene isomerization on Sycamore

**A variational quantum eigensolver**

A parameterized quantum circuit, with properly prepared initial states and with the aid of a classical co-processor, approximates the wave function of a chemical compound. The circuit corresponds to the one used for six-qubit Hartree-Fock calculation (2).

**Google Sycamore**

The schematic shows the superconducting qubits and the adjustable couplers that connect them.

**Quantum circuit**

The input states of the wave function are initialized as $|0\rangle$ or $|1\rangle$. Each box with a parameter $\theta$ is a rotation gate that is compiled into two two-qubit gates and three single-qubit gates.

**Classical calculation**

The outcomes are fed into a classical computer that trains the parameters to learn the ground state of the system. The classical parameters that determine the wave function are fed back to the quantum computer, and the calculation is rerun.

---

**Summary**

![Energy landscape](image)

**Fig. 3. QVE performance on distinguishing the mechanism of diazene isomerization.** Hartree-Fock curves for diazene isomerization between cis and trans configurations. TS1 and TS2 are the transition states for the in-plane and out-of-plane rotation of the hydrogen, respectively. The yellow arrows on TS1 and TS2 indicate the corresponding reaction coordinate. The solid curve is the energy obtained from optimizing a 3D-qubit problem generated by freezing the core orbitals generated from two self-consistent field cycles. The transparent lines of the same color are the full 12-qubit system, indicating that freezing the lowest two levels does not change the characteristics of the model chemistry. Nine points along the reaction pathways are simulated on Sycamore by using QVE. We allowed the optimizer 3D iterations for all points, except for fifth and sixth points from the left of the in-plane rotation curve, for which we allowed 60 steps. The error bars for all points were computed by estimating the covariance between simultaneously measured sets of 1-RDM elements and resampling those elements under a multivariate Gaussian model. Energies from each sample were tabulated, and the standard deviation is used as the error bar. No purification was applied for the computation of the error bar. If purification is applied, the error bars become smaller than the markers. Each basis rotation for diazene contains 50 $\sqrt{\text{SWAP}}$ gates and 80 Rz gates.
Quantum Advantage
Quantum Advantage

• When developing algorithms for quantum computers, the goals tend to focus on either

  • Algorithms with the potential for dramatic speed increases
  • Algorithms which cannot be solved on a classical computer

• The latter, is referred to by a term coined by John Preskill (Caltech), defined as follows

  • **Quantum supremacy** = “the point when quantum computers can do things that classical computers can’t, regardless of whether those tasks are useful.”

• An alternative term in use, which avoids political connotations is quantum advantage, so I’ll stick to that here today
Quantum Advantage

- In Oct 2019, Google published a paper in Nature claiming they had achieved quantum advantage by solving a problem in 200s on Sycamore that would take Summit 10k years.

- The problem: sampling numbers from a pseudo-random quantum circuit.

---

**Article**

**Quantum supremacy using a programmable superconducting processor**


The promise of quantum computers is that certain computational tasks might be executed exponentially faster on a quantum processor than on a classical processor. A fundamental challenge is to build a high-fidelity processor capable of running quantum algorithms in an exponentially large computational space. Here we report the use of a processor with programmable superconducting qubits to create quantum states on 53 qubits, corresponding to a computational state-space of dimension 2^53 (about 10^16). Measurements from repeated experiments sample the resulting probability distribution, which we verify using classical simulations. Our Sycamore processor takes about 200 seconds to sample one instance of a quantum circuit a million times—the benchmarks currently indicate that the equivalent task for a state-of-the-art classical supercomputer would take approximately 10,000 years. This dramatic increase in speed compared to all known classical algorithms is an experimental realization of quantum supremacy for this specific computational task, heralding a much-anticipated computing paradigm.
For $m=20$, it takes 200s to obtain a million samples on the quantum processor.
Response to Google’s Claim

“We argue that an ideal simulation of the same task can be performed on a classical system in 2.5 days”

In this work, we present a tensor network-based classical simulation algorithm. Using a Summit-comparable cluster, we estimate that our simulator can perform this task in less than 20 days. On moderately-sized instances, we reduce the runtime from years to minutes, running several times faster than Sycamore itself.
Simulating the Sycamore quantum supremacy circuits

Feng Pan\textsuperscript{1,2} and Pan Zhang\textsuperscript{1,*}

\textsuperscript{1}Institute of Theoretical Physics, Chinese Academy of Sciences, Beijing 100190, China
\textsuperscript{2}School of Physical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China

We propose a general tensor network method for simulating quantum circuits. The method is massively more efficient in computing a large number of correlated bitstring amplitudes and probabilities than existing methods. As an application, we study the sampling problem of Google's Sycamore circuits, which are believed to be beyond the reach of classical supercomputers and have been used to demonstrate quantum supremacy. Using our method, employing a small computational cluster containing 60 graphical processing units (GPUs), we have generated one million correlated bitstrings with some entries fixed, from the Sycamore circuit with 53 qubits and 20 cycles, with linear cross-entropy benchmark (XEB) fidelity equals 0.739, which is much higher than those in Google's quantum supremacy experiments.

Using our method, employing a small computational cluster containing 60 graphical processing units (GPUs), we have generated one million correlated bitstrings with some entries fixed, from the Sycamore circuit with 53 qubits and 20 cycles, with linear cross-entropy benchmark (XEB) fidelity equals 0.739, which is much higher than those in Google’s quantum supremacy experiments.

Opinion

Why Google’s Quantum Supremacy Milestone Matters

The company says its quantum computer can complete a calculation much faster than a supercomputer. What does that mean?

By Scott Aaronson
Dr. Aaronson is the founding director of the Quantum Information Center at the University of Texas at Austin.

Oct. 30, 2019
Quantum Advantage using Boson Sampling

- In December 2020, a team from USTC claimed that they had obtained quantum advantage using boson sampling.

- The problem: calculate the probability distribution of a system of many bosons (photons).

- Classically, the solution time increases exponentially (#P-hard).

- Simulate the process directly by allowing photons to interfere and sampling the distribution.

- Information encoded in positions and polarization.

- 200 seconds vs 2.5 billion years.

- Clear quantum advantage; however, the computer can only solve one problem.
Conclusion

- Developing quantum algorithms is very challenging
  - Only a handful of algorithms exist, still all listed on a single webpage
- Introduction to some of the most important algorithms
  - Quantum Fourier transform
  - Shor’s factoring algorithm
  - Grover’s and Grover-Long search algorithms
- Also covered a promising (likely early) use case for quantum computers
  - Quantum simulation
- Quantum advantage has long been a target for quantum computation
  - Can be argued that it was achieved in 2019/2020
- Finally, a few slides about programming languages and SDKs for quantum computers and quantum error correction